Remarks

The above amendments and these remarks are in reply to the Office Action mailed April 16, 2008.

I. Summary of Examiner's Rejections

Claims 1, 2, 5-9, 12, 13, 16-20, 29, and 30 were rejected under 35 U.S.C. 103(a) as being unpatentable over Kampe et al. (U.S. Pat. No. 6,854,069).

II. Summary of Applicants' Response

Expedited allowance of the Application is respectfully requested.

III. Response to Rejections

Response to the Office Action's Assertion of Mere Duplication and Routine Skill

The Office Action conceded that Kampe did not disclose the invention described in Claim 1, including the requirement that "additional plugins may be plugged into the resource interface for other resource types." The Examiner had previously commented on the novelty of Applicants' claimed invention during the Examiner interview of December 18, 2006.

However, the Office Action argued that the claimed invention would have been mere duplication of the essential working parts of Kampe and would involve only routine skill in the art. The Office Action cited *St. Regis Paper Co. v. Bemis Co.*, 193 USPQ 8 (7th Circuit, 1977) for this proposition. In *St. Regis Paper Co. v. Bemis Co.*, a patent application was found obvious when one bag was placed inside another bag to strengthen the bag, the claimed invention being

double-bagging to create a stronger bag. There is clearly a distinction between the idea of double-bagging and the invention described in claim 1.

The advantages of the invention defined by claim 1 include that it provides a uniform high availability framework for use in a cluster that includes different resource types. One of the problems with currently available clustering systems is the need to integrate the cluster framework with its software applications by providing a set of application-specific callbacks. These application-specific callbacks are needed to allow adequate control and monitoring of the software applications running on the cluster. Examples of callbacks include application scripts, DLL's, and regular compiled/executable code. In a traditional cluster, each additional type of application server would require their own specific callback. However, using the present embodiment, each resource instance (e.g. each software application server instance) can be of a particular resource type (e.g. application servers and transaction processing systems). Plugins implementing the resource API encapsulate their resource type-specific behavior, and isolate the Java-based cluster server from that behavior. The plugins then provide the mapping between framework's resource management abstractions and the resource type-specific way of realizing the particular functionality, without requiring their own specific callback on the cluster server.

Further Discussion of the Invention of Claim 1 and Kampe

The invention of claim 1 defines that the resources (the application servers and transaction processing systems) can be grouped by resource type within a pool of resources. A Java-based cluster server allows access to the set of resources using a resource interface, which in turn provides an abstraction layer and allows the Java-based cluster server to receive uniform

requests from the software application and communicate the requests to the resources. The system includes a plugin for each resource type, wherein each plugin implements a resource API to encapsulate its particular resource type-specific behavior and to isolate the cluster server from that behavior, while at the same time providing access to its pool of resources. Applicants respectfully submit that these features are not disclosed or suggested by the cited references.

Kampe discloses a system and method for achieving high availability (HA) in a networked computer system. As disclosed therein, the networked computer system includes nodes that are connected by a network. The method includes using high-availability-aware components to represent hardware and software in the networked computer system, managing the components to achieve a desired level or levels of redundancy, and monitoring the health of the networked computer system, including the health of the components and the nodes. (Column 2, lines 38-43). An HA system may be comprised of many independent "managed components." Such components may include: (1) hardware "field replaceable units" ("FRUs"), which can be powered on and off, removed, or replaced without affecting the operation of other hardware components, and (2) independent software functions ("logical FRUs"), which can be shut-down, restarted, or upgraded without affecting other software in the system. Components may be "CPU nodes," each of which is running an instance of an operating system. Using "CPU nodes," the system enables individual nodes to be rebooted with little or no effect on other nodes. Further, hardware components may be distributed across multiple independent shelves and buses so the failure of a single fan, power-supply, CPU, or even a backplane does not take out the entire cluster or networked computer system. (Column 6, lines 35-62). A hot-swap manager ("HSM") may provide a higher-level service that runs above hot-swap bus managers, such as

cPCIs, and managers for other types of FRU interconnects, such as an application that watches for payload cards to come alive on dedicated serial links or join an ethernet. The underlying bus managers are typically responsible for detecting and identifying newly inserted FRUs. The HSM may be responsible for deciding what to do with the new device. The HSM preferably knows about types of supported FRUs through code and/or configuration data, for example. For each supported FRU type, the HSM preferably knows which component managers are to be instantiated to make the FRU useful. (Column 16, lines 44-61).

In the Office Action it was submitted that Kampe discloses that components may be application servers, and that a plurality of plugins can be plugged into system to provide a mapping between the systems resource management functions and resource type-specific (i.e. application server-specific functionality). However, the above description appears to suggest that, in Kampe, while each node may be running an instance of a particular *operating system*, there doesn't seem to be any discussion of, for example, multiple types of operating systems, or of multiple types of different software application servers running on those operating systems. By contrast, as defined by claim 1, the system therein allows a client application to access a set of resources of various resource types, including *application servers and transaction processing systems*. The system includes a plugin for each *type* of application server and transaction processing system, and the plugin implements a resource API to encapsulate its particular application-server-specific behavior from the Java-based cluster server, so that the Java-based cluster server can instead provide a uniform interface to the software application.

Furthermore, as disclosed by Kampe, an HSM is responsible for deciding what to do with a new device, in that the HSM preferably knows about types of supported FRUs through code

and/or configuration data, and preferably knows which component managers are to be instantiated to make the FRU useful. The above description appears to suggest that, in Kampe, the system must incorporate specific code into the server to support each FRU type, which appears analogous to incorporating application-specific callbacks in the server itself. However, as defined by Applicants' Claim 1, a plurality of plugins are plugged into a resource interface, which in turn provides a set of application-specific callbacks, but otherwise isolates the Java-based cluster server from the application-specific behavior of the different software application servers and resources.

In view of the above comments, Applicants respectfully submit that claim 1 is neither anticipated by, nor obvious in view of the cited references, and expedited allowance thereof is respectfully requested.

Claims 2, 5-9, 12, 13, 16-20, 29, and 30

Independent claims 12 and 29 should also be patentable for similar reasoning. Dependent claims 2, 5-9, 13, 16-20, and 30 should be patentable because they depend on patentable independent claims. The dependent claims also add their own limitations which render them patentable in their own right.

IV. Conclusion

In view of the above amendments and remarks, it is respectfully submitted that all of the claims now pending in the subject patent application should be allowable, and reconsideration

thereof is respectfully requested. The Examiner is respectfully requested to telephone the undersigned if he can assist in any way in expediting issuance of a patent.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 06-1325 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: July 10, 2008

By: /Thomas K. Plunkett/
Thomas K. Plunkett
Reg. No. 57,253

Customer No.: 80548
FLIESLER MEYER LLP
650 California Street, 14th Floor
San Francisco, California 94108
Telephone: (415): 362-3800